Handbuch
User Manual
logico-hs-io

Version 1.1
Artikel-Nr./Part-No.: 10495

List of Content

1. Logico-hs-io System
   1.1 Boards - System interfaces 5
   1.2 logico-io bus - General 5
   1.3 Hardware 5
   1.4 Software 6

2. Digital boards
   2.1 logico-hs-io-dig - General 8
   2.2 Hardware 9
   2.3 Software 13

3. Analog boards
   3.1 logico-hs-io-ana - General 14
   3.2 Hardware 15
   3.3 Software 18

4. Software for all boards 19

1 logico-hs-io System

logico-hs-io system is a modular DIN rail mounted interface for I/O signals over RS-422 (Fig 1). In addition to the serial RS-422 (a) the system can be accessed via Ethernet (b) and USB (c). The Ethernet link uses the TCP/IP protocol. The system is composed of required combination of analog and digital IO boards and may contain one Basic Module for TCP/IP or USB Controller interface.

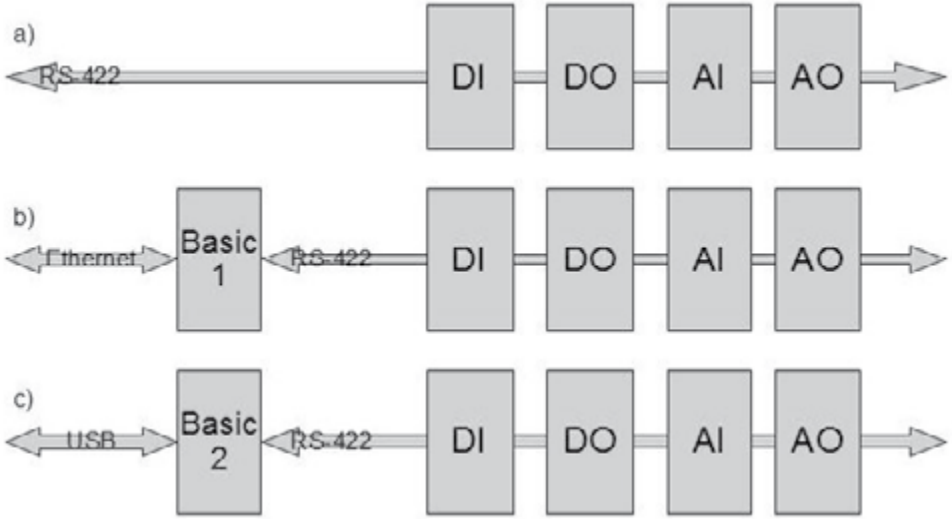

Fig.1 logico-hs-io system

Controller

Controller is a software concept defined by a protocol described in a document “IO Board System - Chain Protocol.pdf”. The Controller may be located at any place; in many cases it will be a PC. The Controller may communicate with the IO Board System in one of three ways: TCP/IP, USB or RS-422:

- The native RS-422 communication does not require any hardware (Basic) interface boards.
- The TCP/IP communication requires thelogico-hs-io bus interface board.
- The USB communication requires a converter USB/RS-422.

1.1 Boards - System interfaces

There are two Basic boards for TCP/IP and USB system interface:

- The logico-hs-io bus which is a TCP/IP to RS-422 interface.
- The board BasUSB which is a USB to RS-422 interface (in design).

1.2 logico-io bus - General

The logico-hs-io bus board is the interface between a Controller and input/output (I/O) boards in the IO Board system. The board can be connected with the Controller via Ethernet (TCP/IP.) Thelogico-hs-io bus board connects to other I/O boards via RS-422 bus.

Thelogico-hs-io bus board organizes the communication between Controller and I/O boards. It passes the TCP/IP messages coming from Controller to the RS-422 bus where they are catched by I/O boards. It passes the RS-422 messages coming from the I/O boards to the TCP/IP channels going to Controller.

The board is supplied in a DIN Rail enclosure.

Only one logico-hs-io bus board can be connected to the RS-422 bus.

1.3 Hardware

Ethernet: ISO 8802.3 100BASE-T or 10BASE-T
RS-422 bus: 115200 baud.
Power supply: 24V
Enclosure size (WHD): 53x90x58 mm
Label:logico-hs-io bus-01
Connectors:

- Power supply 24V: two screw terminals 24V and 0V.
- Ethernet: 1 x RJ45.
- RS-422 bus: PHOENIX CONTACT COMBICON.

1.4 Software

The communication protocol is defined for message exchange between a Controller and one or more I/O boards. The BasEth board passes only messages between the Controller and the I/O boards: it does not process them apart from media conversion.
The BasEth board has 2 TCP/IP ports:

- 58001 for communication with the BasEth board.
- 58003 for communication with I/O boards.

Communication speed on the TCP/IP side: 100 messages / sec.
The communication via the port 58001 covers the query / setting of TCP/IP parameters like:

- IP address
- MAC address
- Software version
- Compilation time.

The communication via the port 58003 realizes the message exchange between the Controller and the I/O boards connected to the RS-422 bus. It is a query channel where the IO boards respond to Controller requests. In addition this port conveys events generated in the IO boards to the Controller.
The following functions are provided for the communication with the BasEth board:

bool Connect(const string ip)

This function connects the program to the boards using two tcp/ip channels: Basic and Control. Both channels use the same IP address but different ports. Ports has the values: 58001 (Basic) and 58003 (Control) and cannot be changed.

- The Basic channel is for communication with the Ethernet interface (Basic board).
- The Control channel is for communication with I/O interfaces (Control boards).
- The parameter ip is a string that must be equal to the IP address of the board. The default IP address of the board is **169.254.19.63**. The IP address can be changed using the function SetIPAddress. If the parameter ip is different than the actual IP address of the board the connection will fail and there is no way to get the address from the hardware. A button on the board allows setting of the default IP address.

The function returns true if the connection has been successful to both channels, otherwise false.

void Disconnect()

Disconnects the program from the boards: both (Basic and Control) channels are disconnected.

bool GetMACAddress(string &rcv)

The function reads the MAC address of the Basic interface board.

The function returns true if the MAC address has been read, otherwise false.

bool GetVersion(string &rcv)

The function reads the version of the Basic interface board. The string representing the version is something like „ B:RS422_TCP SW:0.02E”

bool SetIPAddress(string snd)

The function sets a new IP Address. The new address becomes valid after restart of the board. See also comments about the function Connect().
The function returns true if the IP address has been set, otherwise false.

void EnableLog(bool enable)

The log file logs all messages passed between a program and boards.
Note that the C# version is different:

string EnableLog(bool enable) (for C#)

If the parameter enable is true the function creates a log file with the name logRandom-Numer.txt where the RandomNumber is a value between 0 and 1000, otherwise it closes the log file if it exists. This arrangement allows creation of several log files in the same directory if many application programs are running simultaneously.
The log file logs all messages passed between a program and boards.
The log file returns the file name if the file operation has been successful, empty string otherwise.

2 Digital boards

Digital boards handle 2-state signals that are named as: low – high, or false – true. They are represented as voltage levels which nominal values are: 0 and 24V. There are digital inputs and digital outputs.

2.1 logico-hs-io-dig - General

The logico-hs-io-dig board is an i/o board in the IO Board system. It communicates with the Controller (possibly via the Basic board) using the RS-422 bus. It can be used as a stand-alone board connected directly to the Controller via an RS-422 interface. The board is supplied in a DIN Rail enclosure.

- An input detects a voltage level (nominal value 24V).
- An output generates a signal that may be either low or high value.
- The number of boards connected to the system bus is limited to 200 (addressing).

2.2 Hardware

Power supply: 3.3V (normally supplied by the Basic board via the RS-422 bus), 20mA.
Number of inputs: 4

Input features:

- Inputs are isolated from the system and from each other (1000V)
- Input states are signaled by LEDs:
  - An input voltage below +9V (or negative) is signaled as low (LED switched off)
  - An input voltage above 9 V (depends on load: it may light not until) is signaled as high (LED green)
  - A reverse input voltage is allowed (LED orange)
  - Max input voltage: +/- 31V

Number of outputs: 2

Output features:

- Outputs are isolated from the system and from each other (1000V)
- Output states are signaled by LEDs:
  - An output below +9V is signaled as low (LED switched off)
  - An output voltage above 9 V (depends on load: it may light not until 15V) is signaled as high (LED green)
  - An overload of output current is signaled by a red LED
  - Outputs can be of sink or source type

Enclosure size (WHD): 53x90x58 mm

Label: logico-hs-io-dig

Connectors:

- Digital inputs: Din+, DGNDn- (n = 0..3) and 2 LEDs are assigned to each input.
- Digital outputs: VCn+ (internal power supply), Cn (sink output), En (source output) and GNDn, where n = 0, 1 and 2 LEDs are assigned to each output.
- RS-422 bus: PHOENIX CONTACT COMBICON.

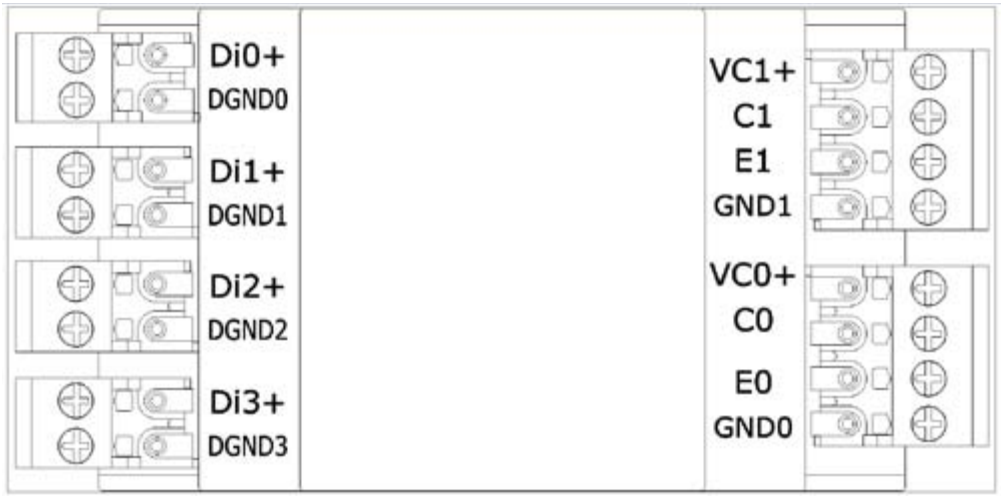

Fig.4 logico-hs-io-dig

Using inputs

The Input voltage is applied between terminals DGNDn and Din+.

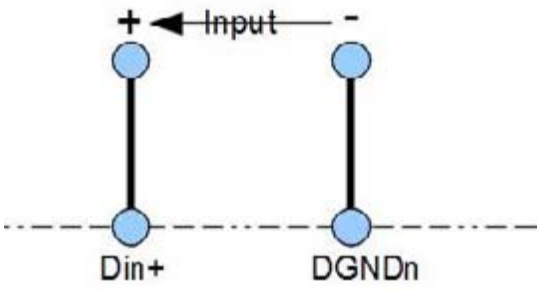

Fig.5 using digital outputs

Using outputs

Source output using internal power supply

To get it working the terminal Cn has to be connected with the terminal VCn (internal power supply). If switched on the Output generates high voltage.

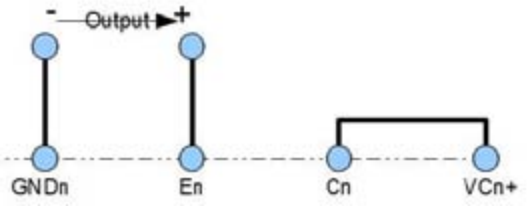

Fig 6: Using digital outputs as source with internal power supply

Source output using external power supply

To get it working the external 24V has to be applied between terminals GNDn and Cn.

The Output generates high voltage if switched on.

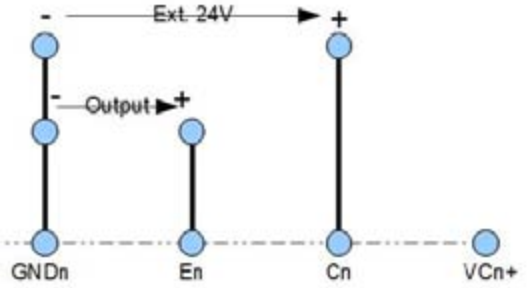

Fig 7: Using digital outputs as source with external power supply

Sink output using internal power supply
To get it working the terminal GDNn has to be connected with En. The output is then between terminals Cn and VCn+ (internal power supply). If switched on the Output generates high voltage (Cn is on GNDn).

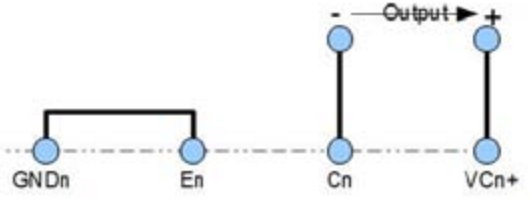

Fig 8: Using digital outputs as sink with internal power supply

Sink output using external power supply
To get it working the minus (-) of an external power supply has to be applied to the terminal En. The output is then between the plus (+) of the external power supply and the terminal Cn. If switched on the Output generates high voltage (Cn is on GND).

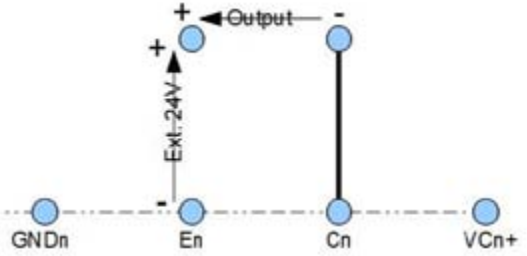

Fig 9: Using digital outputs as sink with external power supply

Testing input and outputs
The digital inputs and outputs may be connected with each other. For instance, the digital output Do0 working as a source output with internal 24V can be tested using an digital input Di3.

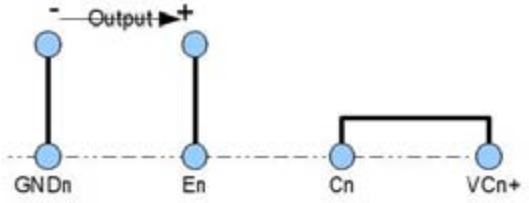

Fig 10: Testing a digital input using source digital output

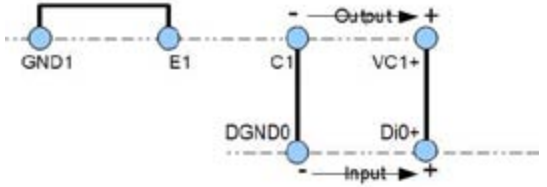

Fig 11: Testing a digital input using sink digital output
Another example shows the digital output Do1 working as a sink with internal 24V connected with digital input Di0.

2.3 Software

The digital inputs can be polled or can be set up for events. They are polled using the command GetInput whose response contains all digital inputs. They are set for events using the command SetEventMask and collected using the broadcast command GetEvents.
The “0” logic level corresponds to voltage above 9V and the “1” level to a low level below 9V, measured at the input connector. An erroneous connection on the input connector is signaled as a „0“ level and signaled by the LED as red.
The digital outputs are set using the function:

ioBoardResult SetDigitalOutput(unsigned char boardId, bool outMask[], bool outValue[])

The function sets output values (low, high) in the board specified by the parameter boardId. The output values are specified by two parameters:

- outMask which defines which output to set
- outValue which defines the value.

This is the only logico-hs-io dig board specific function. For other functions see “Software for all boards“.

3. Analog boards

Analog boards handle analog signals that are currents. Inputs measure current in the range 0…20mA and outputs generate current in the range 4…20mA.

3.1 logico-hs-io-ana - General

The logico-hs-io-ana board is an analog input/output board in the IO Board system. It communicates with the Controller (possibly via the Basic21 board) using the RS-422 bus. It can be used as a stand-alone board connected directly to the Controller via an RS-422 interface. The board is supplied in a DIN Rail enclosure.
Both the analog inputs and the analog output are of current type.
The number of boards connected to the system bus is limited to 200 (addressing).

3.2 Hardware

Power supply: 3.3V (normally supplied by the Basic board via the RS-422 bus), 20mA
24V (for analog output)

Analog inputs:

- Number of inputs: 4
- Range: 0 to 20 mA
- Resolution: 24 bits (up to 23 bits noise-free)
- Accuracy: 0.1% (without calibration and correction)
  0.01% (with calibration and correction)
- No missing code: 24 bits
- Input resistance: 250 ohm
- Max input current: 20 mA

Analog output:

- Number of outputs: 2
- Range: 4 to 20 mA
- Resolution: 12 bits
- Accuracy: LSB
- Guaranteed Monotonicity
- Max load resistance: 600 ohm
- The output is isolated from the inputs (1000V).

Enclosure size (WHD): 53x90x58 mm
Label:logico-hs-ioana-01
Connectors:

- Analog inputs: to each input are assigned two screw terminals Ain+ and Ain- where n = 0...3. In addition there is a common screw AiCOM. This should be wired to a suitable local analog ground or common potential, and also to unused inputs AIn- pins.
- Analog outputs: there are two outputs. The output 0 one has two screw terminals Ao0+ and AGND. The output 1 has two screw terminals: Ao1+ and +24V-1.
- RS-422 bus: PHOENIX CONTACT COMBICON.

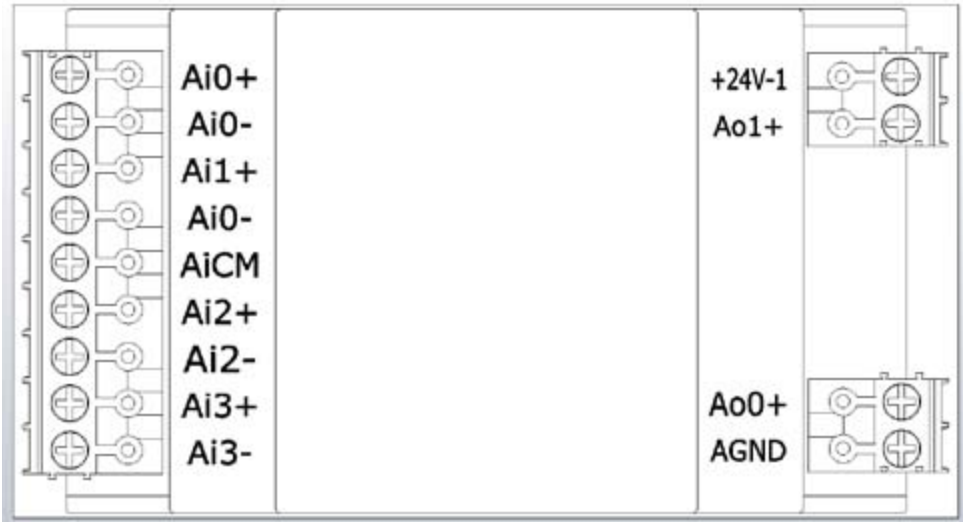

Fig.13 logico-hs-io-ana

Using inputs

The source of the Input current is applied between terminals DGNDn and Din+. In addition a separate connection of the Common potential to AiCM is required. As the inputs are of differential type, care must be taken while connecting a current source:
If the current source is connected locally to ground, then a separate wire has to connect the source local ground with the Common potential AiCM.
If the current source is floating, the negative input terminal Ain must be connected with the Common potential AiCM (see “Fig 17: Testing analog inputs using analog output Ao0“).

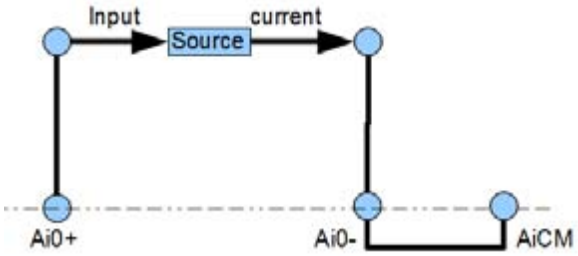

Fig 14: Using analog (current) input

Using outputs

- The output 0
  This Output generates current 4...20mA using the internal 24V power supply. The load is connected between the output terminals Ao0+ and AGND.

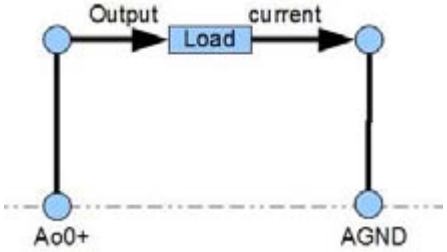

Fig 15: Using analog (current) output0

The output 1
To get it working an external 24V power supply has to be applied to the terminal +24V-1. The load is then connected between the terminal Ao1+ and the minus (-) of the 24V power supply. The Output generates current 4...20mA.

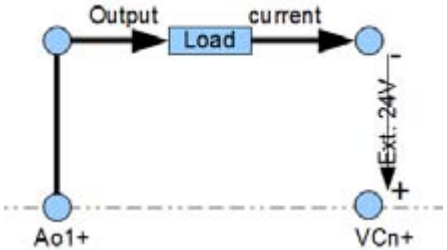

Fig 16: Using analog (current) output1

Testing inputs and outputs

Both outputs can be connected with inputs which gives simple circuit to test them.
Testing output Ao0

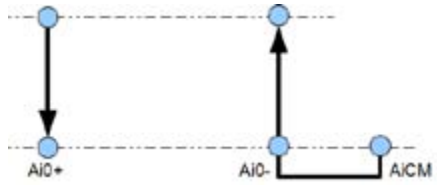

Fig 17: Testing analog inputs using analog output0

Fig 18: Testing analog inputs using analog output1

3.3 Software

The analog inputs can be polled or can be set up for events. They are polled using the function GetInput whose response contains all analog inputs. They are set for events (each input separately) using the function SetEventMask and SetEventThreshold and collected using the broadcast function GetEvents.
There are two Ai4Ao2 board specific functions:

ioBoardResult SetAnalogCurrentOutput(unsigned char boardId, unsigned char outNum, double anaValue)

which sets the current value defined by a parameter anaValue on the analog output outNum in the board boardId.

ioBoardResult GetAnalogCurrentOutput(unsigned char boardId, unsigned char outNum, double &anaValue)

which reads the current value anaValue on the analog output outNum in the board boardId.

For other functions see “Software for all boards“.
The board provides additional functionality Ao= f(Ai). It has the PID regulator (see a separate document “PID Regulator.pdf”.

Other boards
Any combination of digital and analog inputs and outputs interfaces can be combined on a board. This user specific boards are designed according to client specifications. Ask for conditions and deadlines.

4. Software for all boards

There are a several functions that can be applied to any board:

ioBoardType GetBoardType(unsigned char boardId)

The function reads for the board boardId its type defined by the enumeration ioBoardType.

ioBoardResult GetStatus(unsigned char boardId, string &st, RcvStatus s&tatus)

The function reads the status of the board boardId. The status contains information defined by the structure RcvStatus. In addition the status is returned as a string ready for display.

ioBoardResult GetHardware(unsigned char boardId, string &st, MyHardware &hardware)

The function reads some hardware features of the board boardId. The hardware features contain information defined by the structure MyHardware. In addition the hardware fetures are returned as a string ready for display. This information is normally not interesting for a typical user interface.

ioBoardResult GetBoardVersion(unsigned char boardId, string &st)

The function reads the version of the board boardId in a form of a string. For instance for the board Ai4Ao2 it may be: “”.

ioBoardResult GetInputs(unsigned char boardId, double anaVal[], bool digVal[])

The function reads input values for the board boardId. The result is available either as an array of (double) values anaVal for a board with analog inputs or as an array of (boolean) values digVal for a board with digital inputs.

ioBoardResult SetEventMask(unsigned char boardId, ioBoardType boardType, bool mask[])

The functions sets the events mask for the board boardId and defined type boardType. The mask must be defined as a boolean array (true value for setting the event, false otherwise). The array index corresponds to the input number, for instance mask[0] defines Ai0.

ioBoardResult SetEventThreshold(unsigned char boardId, double thrValue[]

The functions sets the threshold values for all inputs of the the board boardId. An event will be send if abs(value change) >= threshold. The boards must of of type Ai4Ao2or Ai4Ao1. The thrValue must be defined as a double array. The array index corresponds to the input number, for instance thrValue[0] defines threshold value for Ai0.

ioBoardResult GetEventMask(unsigned char boardId, bool mask[])

The function reads the event mask for the board boardId. The mask is read as a boolean array. The array index corresponds to the input number, for instance mask[0] defines Ai0.

There is a broadcast function:

ioBoardResult GetEvents(unsigned char &board, int &arrSize, AiEvent anaInput[], DiEvent digInput[])

This function reads events defined by event masks from all boards. The events are delivered as arrays of analog values anaInput for boards with analog inputs and arrays of boolean values digInput boards for boards with digital inputs. One GetEvents query may produce several answers (see “Events“ for detailed description).

The software uses a few enumerations:

All functions return a value of the enumeration type:

```
enum ioBoardResult
{
  NotOK, // request failed
  OK, // request succeeded
  Pending, //still data to receive
  ValueTooLarge, // the required value is too large: it has been set to a default max value 20mA
  ValueTooSmall, // the required value is too small: it has been set to a default min value 4mA
  BoardNumberTooLarge, // the board number is too large
  BoardTypeWrong,  // unknow board type
  InputNumberTooLarge, // the input number is too large
  OutputNumberTooLarge, // the output number is too large
  UnknownState /// unknown Controller state
}
```

The board type is defined by an enumeration:

```
public enum ioBoardType
{
  Unknown, // unknown type
  Ai4Ao1, // board response: „AI4 AO1 V01“
  Ai4Ao2, // board response: „AI4 AO2 V01“
  Di6, // board response: „DI6 V01“
  Di4Do2 // board response: „DI4DI2 V01“
}
```

The board status is defined by an enumeration:

```
public struct RcvStatus
{
  unsigned char flags; // tem
  unsigned short over_flow; // overflow counter
  unsigned short chk_err; // check sum error counter
  unsigned short start_err; // start byte error counter
  unsigned short rec_frames; // frame received counter
  unsigned short chk_sum; // last check sum value
}
```

The hardware features are defined by a structure:

```
public struct MyHardware
{
  unsigned char hw_reserved;
  double hw_Temp;  // temperature in MSP controller
  double hw_Vcc; // board power supply (nominal value 3.3V)
  double hw_Temp30;
  double hw_Temp85;
  unsigned short hw_Tick; // 50 Hz pulses
 }
```

Writing application software - General

The application software which accesses the boards performs a continuous query of input signals and sets output signals if required. A creation of a software for ioBoards is based on a DLL which provides all functions described in this document. The communication with boards in these functions is not interruptable that means they may be called simultaneously (in parallel, for instance in different threads) and their never collide due to internal synchronization mechanism that are built in (transparent for a programmer).
Several versions of a DDL are available: for C++ and C# programming languages and for Ethernet interface and direct RS-422 (covers also USB) interface. The description in this document applies to the C++ version. The C# version may differ in some details from the C++ version. By programming use the appropriate version DLL documentation.

Events

IoBoard protocol defines a concept of an Event as an input change To become an event the input has to be configured as an event source using the function SetEventMask. In addition, the threshold value for analog inputs (Ai4Ao2) should be defined using the function SetEventThreshold.
The function GetEvents is then used to collect events from all boards in the system. The answer to the GetEvents function contains events from all boards as separate messages (frames). The function processes an event message from one board. If the processed event message has been the last one the function returns the value OK, otherwise the value Pending. It means that the function has to be called in a loop where the function is called several times until all event messages are received.

Polling and Events

Data acquisition requires continuous reading of inputs. Having available only polling mechanism the polling frequency is determined by inputs which have to be read most frequently. This requirement means that polling must run in general too fast consuming most of the CPU time. A concept of Events allows optimization of queries by using several threads with different polling loops. The simplest system may contain:
A thread with a fast polling loop calling Get Events, where only inputs which must be actualized very frequently are defined as events.
Several threads with a slow polling loop calling GetInputs for each board separately.

Example of a console application

The Manual is accompanied by a demo program ioBoardCppTest which is a console application demonstrating the use of all functions available in the DLL. The program is written using the C++ DLL. Alternatively, the C# DLL is provided. Each DLL is provides in two variants: for Ethernet and RS-422 interface.

www.tci.de

tci GmbH • Ludwig-Rinn-Straße 10-14 • D-35452 Heuchelheim/Gießen • Tel: +49 6 41-9 62 84-0 • Fax: +49 6 41-9 62 84-28 • vertrieb@tci.de